Problem 1

We have seen that a grammar where all productions are of the form:

\[ A \rightarrow aB, \]
\[ A \rightarrow c \]

(where \( A, B \) non-terminals, and \( a, c \) terminals) defines a regular language, and that from the grammar we can build directly an FSA that recognizes the corresponding language. Show that if all the productions have the form:

\[ A \rightarrow Ba, \]
\[ A \rightarrow c \]

such a grammar also represents a regular language.

Solution:

A grammar \( G = (V, \Sigma, R, S) \) where all the productions in \( R \) are of the form \( A \rightarrow aB \) and \( A \rightarrow c \) is called left grammar.

A grammar \( G = (V, \Sigma, R, S) \) where all the productions in \( R \) are of the form \( A \rightarrow Ba \) and \( A \rightarrow c \) is called right grammar.

We have seen that a left grammar define a regular language. We want to show that a right grammar define a regular language, too.

Let \( G = (V, \Sigma, R, S) \) be a left grammar. Consider the grammar \( G' = (V, \Sigma, R', S) \), where \( R' \) is obtained reverting all the productions in \( R \) i.e., \( R' = \{ A \rightarrow w | A \rightarrow w^r \text{ is in } R \} \).

Note that \( G' \) is a right grammar, since it was obtained reverting all the productions of the left grammar \( G \). Moreover, it holds that \( L(G) = L(G')^r \).

We already proved that the language generated by a left grammar is a regular set: thus, \( L(G') \) is a regular set. We also know that the set of regular languages is closed under reversal: hence, \( L(G')^r \) is also a regular set. Since \( L(G')^r = L(G) \), we proved that any right grammar generates a regular language.

\[ \square \]
Problem 2

Describe the language generated by the following grammars (upper-case letters are non-terminals, lower case terminals):

1. \( S \rightarrow SaS \mid b \)

2. \( S \rightarrow aSb \mid bSa \mid \epsilon \)

Solution:

1. \( L_1 = \mathcal{L}(b(ab)^*) \), i.e. \( L_1 \) is the language described by the regular expression \( b(ab)^* \) which is equivalent to \( (ba)^*b \).

2. Given a string \( w \), let’s denote with \( w^R \) the reverse of \( w \) (e.g. if \( w = bba \) then \( w^R = abb \)). Besides, given a string \( w \), let’s denote with \( \bar{w} \) the “complement” of \( w \) obtained from \( w \) by swapping \( a \)’s with \( b \)’s (e.g. if \( w = bba \) then \( \bar{w} = aab \)). Now, we can define our language over the alphabet \( \Sigma = \{a, b\} \) as:

\[
L_2 = \{ z \in \Sigma \mid \exists w \in \Sigma : z = w\bar{w}^R \}
\]
Problem 3

The following grammar is not regular but it generates a regular language. Write the regular expression that describes it:

\[ S \rightarrow SSS \mid a \mid ab \]

Solution:

First, observe that the only production that introduce non-terminals in the sentential form is the production \( S \rightarrow SSS \). Thus, for any derivation we can rearrange the order in which the productions are applied so that all the applications of \( S \rightarrow SSS \) came first, followed by application of \( S \rightarrow a \mid ab \), as needed.

Then, we consider the set of sentential form derivable just from the production \( S \rightarrow SSS \). We can notice that after \( t \) derivations, we have exactly \( 3 + 2(t - 1) = 1 + 2t \)'s in the sentential form; thus, the set of sentential forms obtainable with application of the production \( S \rightarrow SSS \) is

\[ A = \{ S^c \mid c = 1 + 2t \} = \{ S(SS)^c \mid c = t \}. \]

From what said above, it is clear that all the strings generated by the grammar are those obtained by applying the productions \( S \rightarrow a \mid ab \) to a sentential form in \( A \).

Hence, the language generated by the grammar is the language described by the regular expression:

\[(a|ab)((a|ab)(a|ab))^* \]
Problem 4

Show that the following grammar is ambiguous, by producing one string in the language that has two different parse trees:

\[ S \rightarrow aS \mid aSbS \mid \epsilon \]

Solution:
Consider the string \( aaba \) and the two parse trees in the figure below:
Problem 5

Show that the language generated by the grammar:

\[ S \rightarrow aSbS \mid bSaS \mid \epsilon \]

is the set of all strings with an equal number of \(a\)'s and \(b\)'s. You might want to prove this by induction over the length of the string, or by reasoning over the sentential forms that derive a string.

Solution:
We first show that each string generated by this grammar has the same number of \(a\)'s and \(b\)'s.
Proceed by induction over the length \(N\) of the derivation.

- **Case base:** \(N = 1\). The only string derivable with a derivation of length \(N = 1\) is \(\epsilon\) \((S \rightarrow \epsilon)\): it trivially holds that the number of \(a\)'s and \(b\)'s in \(\epsilon\) is the same.

- **Inductive step:** Assume that all the strings \(w\) derivable from \(S\) with a derivation of length at most \(N\) have an equal number of \(a\)'s and \(b\)'s. Consider now a string \(w'\) derivable from \(S\) with a derivation of length \(N + 1\). Clearly, the first production used in the derivation that yields \(w'\) is either \(S \rightarrow aSbS\) or \(S \rightarrow bSaS\). In either cases, the remaining \(N\) productions can be ideally split into two derivations: one that generates a string \(w_1\) from the first \(S\) and another that generates a string \(w_2\) from the second \(S\). By induction, it clearly holds that in both \(w_1\) and \(w_2\) the number of \(a\)'s is equal to the number of \(b\)'s. Moreover, since \(w\) is either \(aw_1bw_2\) or \(bw_1aw_2\), we get that the number of \(a\)'s in \(w\) is equal to the number of \(b\)'s.

To complete the proof, we still need to show that any string \(w\) with an equal number of \(a\)'s and \(b\)'s can be derived from \(S\). Proceed by induction on the length of \(w\).

- **Case base:** \(|w| = 0\), i.e. \(w = \epsilon\). It is clear that \(w\) can be derived from \(S\) \((S \rightarrow \epsilon)\).

- **Inductive step:** Assume that all the strings \(w\) with an equal number of \(a\)'s and \(b\)'s and of length up to \(2N\) are derivable from \(S\). Consider now a string \(w'\) with an equal number of \(a\)'s and \(b\)'s of length \(2(N + 1)\). W.l.o.g. assume that the first symbol in \(w'\) is an \(a\). Let \(2 \leq j \leq 2N + 2\) be the first index such that the substring of \(w'\) from position 1 to position \(j\) has an equal number of \(a\)'s and \(b\)'s. Notice that such \(j\) always exists, since in the worst case \(j = 2N + 2\) will do it.

If \(j = 2N + 2\), then this means that \(w'\) is of the form \(a^N b^N\) and thus can be derived from \(S\) by the derivation \(S \Rightarrow aSbS \Rightarrow aSb \Rightarrow aaSbSb \Rightarrow aaSbb \Rightarrow \ldots \Rightarrow a^N b^N\).

Otherwise (i.e. \(j < 2N + 2\)), notice that, since the symbol at position \(j\) must be a \(b\), \(w'\) can be written as \(w' = aw_1bw_2\), where \(w_1\) and \(w_2\) are both strings of length at most \(N\), each having an equal number of \(a\)'s and \(b\)'s. Thus, by induction, both \(w_1\) and \(w_2\) can be derived from \(S\) from which it follows that \(w'\) can also be derived from \(S\).